TRUSTED PROCESS CLASSES

William L. Steffan
Tracor Applied Sciences, Incorporated
503A Coliseum Boulevard
Montgomery, Alabama 36109
Voice: 334-271-6804 FAX: 334-244-0058
wsteffan@b856s1.ssc.af.mil

Jack D. Clow
SenCom Corporation
6004C East Shirley Lane
Montgomery, Alabama 36117
Voice: 334-277-1972 FAX: 334-277-1932
jclow@b856s1.ssc.af.mil

ABSTRACT

Vendors who develop Trusted Computing Base (TCB) equipped secure operating systems face difficult choices as they design and implement the requisite protection features appropriate to the evaluation class being targeted (e.g., *Labeled Security Protection*, Class B1). On the one hand, vendors seek to meet each and every evaluation class requirement unconditionally, being careful to limit *every possible* opportunity for latent vulnerabilities to occur. However, on the other hand, vendors must not implement their secure product with so many constraints that it loses its competitive advantage and utility as an operating system having general applicability throughout the marketplace. Balancing these conflicting goals often results in the vendor's implementing a *more restrictive* rule set than permitted by theoretical considerations.

Unfortunately, unlike the published criteria for TCB classes themselves[1], developers who implement trusted processes have had to depend on ad hoc experimentally derived guidelines and rules to meet both mission and security requirements simultaneously.

This paper presents a new methodology, derived from the theory of a TCB-equipped operating system and practical experience, to explicitly determine to which of several classes a specific trusted process* belongs, as well citing applicable programming confinement rules to ensure additional risks, if any, will be acceptable.

** A trusted process is a program, module, or algorithm which has extraordinary privilege(s), which if not otherwise strictly controlled and limited, could subvert the security policy in unpredictable ways -- in the extreme, subvert the protection domain provided by the TCB for itself.*

Keywords: Trusted Computing Base, trusted process, mandatory access controls, discretionary access controls, auditing, certification, accreditation.

TRUSTED PROCESS CLASSES

1. **Background.** Vendors who develop Trusted Computing Base (TCB) equipped secure operating systems face difficult choices as they design and implement the requisite protection features appropriate to the evaluation class being targeted (e.g., *Labeled Security Protection*, Class B1). On the one hand, vendors seek to meet each and every evaluation class requirement unconditionally, being careful to limit *every possible* opportunity for latent vulnerabilities to occur. However, on the other hand, vendors must not implement their secure product with so many constraints that it loses its competitive advantage and utility as an operating system having general applicability throughout the marketplace. Balancing these conflicting goals often results in the vendor's implementing a *more restrictive* rule set than permitted by theoretical considerations. For example, while formal TCB theory would permit a *high-clearance* program (i.e., a *subject*) to read a *low-sensitivity* data item (i.e., an *object*), some vendors enforce a *clearance SHALL ALWAYS EQUAL sensitivity* rule rather than the more general *clearance SHALL BE GREATER THAN OR EQUAL TO sensitivity* rule permitted by theory.

Thus, for these and other reasons, TCB-equipped operating systems usually fall short of providing security protection features needed to support *every possible* customer application. In turn, then, software developers who use these TCB-equipped operating systems to meet customer mission-oriented applications face some awkward possibilities:

- Failing to meet the customer's mission requirements since the vendor's TCB-equipped operating system and its protection features prohibits achieving some essential mission functionality.

- Failing to meet the customer's security requirements since the vendor's TCB-equipped operating system does not provide the necessary features to protect some essential mission functionality and/or sensitive information.

- Failing to meet both the customer's mission and security requirements since the vendor's TCB-equipped operating system implements a more restrictive enforcement policy than that permitted by theory.

Classically, the solution has been to incorporate developer-written *trusted processes* so that

- Customer mission requirements will be met unconditionally,

- While accepting some additional risk by permitting local controlled deviations to otherwise globally applicable security policy enforcement rules.

Unfortunately, unlike the published criteria for TCB classes themselves[1], developers who implement trusted processes have had to depend on ad hoc experimentally derived guidelines and rules to meet both mission and security requirements simultaneously.

2. **Purpose.** This paper presents a new methodology, derived from the theory of a TCB-equipped operating system and practical experience, to explicitly determine which of several

classes a specific trusted process belongs to. It also cites applicable programming confinement rules to ensure that any additional risks will be acceptable.

3. **Definitions**. Definitions for key concepts and terms used herein follow.

a. Discretionary Access Controls. *Discretionary Access Controls (DAC)* are rules enforced by the *reference monitor* which provide for need-to-know violation prevention as prescribed by the security policy for the system (i.e., an entity's functional role is a sufficient basis for permitting access to system-protected resources).

b. Integrity Principle. A fundamental underlying assumption of a TCB-equipped operating system that states a *high-integrity* entity (e.g., the *reference monitor*) shall NEVER "trust" assertions made by a *lower-integrity* entity (e.g., an untrusted applications program), but a *low-integrity* entity shall ALWAYS "trust" assertions made by a *higher-integrity* entity.

c. Least Privilege Principle. A fundamental mandate for a TCB-equipped operating system that states *every* entity shall be granted ONLY the MINIMUM privilege(s) essential to perform its assigned function(s) and NO MORE.

d. Mandatory Access Controls. *Mandatory Access Controls (MAC)* are rules enforced by the *reference monitor* which provide for compromise prevention as prescribed by the security policy being enforced for the system (e.g., an entity's clearance is a sufficient basis for permitting access to system-protected resources).

e. Reference Monitor. A *reference monitor* is that portion of a TCB-equipped operating system having exclusive responsibility for enforcing *Discretionary Access Controls* and *Mandatory Access Controls* according to mathematically precise rules.

f. Tranquillity Principle. A fundamental underlying assumption of the Bell-LaPadula formal security model for a *reference monitor* which states that, once identified to the *reference monitor*, the sensitivity level contained in a subject's *sensitivity label* (or an object's *sensitivity label*) shall remain invariant unless explicitly changed under the express control of the *reference monitor*.

g. Trusted Process.

(1) A *trusted process* is a program, module, or algorithm written expressly by a developer that has these characteristics:

- May require over-riding security policy enforcement mechanisms or their underlying assumptions.
- Does not subvert security policy mandated rules except in explicitly controlled ways in a constrained local context.
- **NEVER** enforces globally applicable security policy mandated rules.

(2) A *trusted process* is a program, module, or algorithm which has extraordinary privilege(s), which if not otherwise strictly controlled and limited, could subvert the security

policy in unpredictable ways -- in the extreme, subvert the protection domain provided by the TCB for itself.

4. **Methodology**. This section examines some relevant factors to determine the *Trusted Process Class* for any given trusted process.

a. Trusted Process Observations. In practice, as suggested by themes given in *TCB Subjects - Privileges and Responsibilities* in [2], trusted processes may be granted privileges which over-ride enforcement rules for DAC, MAC, Tranquillity Principle, or any combination thereof. For example:

- To produce an unclassified report about the existence of, but not the value of, some classified fact, a *low-clearance* trusted process subject needs to read, but not reveal, the information content in a *high-sensitivity* object. To do this, the trusted process subject must temporarily over-ride MAC enforcement rules.

- To perform a regrade on some imported file, a *high-clearance* trusted process subject must change the sensitivity label *content* (i.e., the sensitivity level), but not the information content in the object itself, for a *low-sensitivity* object. To do this, the trusted process subject must temporarily over-ride the Tranquillity Principle.

- To perform an emergency recovery action, a trusted process subject must be granted temporary *execution privilege* over-riding DAC enforcement rules.

- To provide standard agency-specified security markings on human readable media, a trusted process subject must intercept and faithfully translate internally coded binary representations for the sensitivity label content. To do this, the trusted process subject must uphold the *Integrity Principle*, while not subverting DAC, MAC, Tranquillity Principle, or other security policy-prescribed rules.

b. Trusted Process Classes. The cited examples suggest a methodology to determine the specific trusted process class for any given trusted process. As Figure 1 illustrates, these trusted process classes can be enumerated according to whether the trusted process must over-ride rules for DAC, MAC, Tranquillity Principle, or combinations thereof.

	Over-Ride Privilege Granted			
Trusted Process Class	Tranquillity Principle	Mandatory Access Controls	Discretionary Access Controls	Permitted Action(s)
0	---	---	---	*Read Only*
1	---	---	Yes	*Read, Write, or both Read & Write*
2	---	Yes	---	
3	---	Yes	Yes	
4	Yes	---	---	
5	Yes	---	Yes	
6	Yes	Yes	---	
7	Yes	Yes	Yes	
	Label	*Content*	*Privileges*	

Figure 1, Trusted Process Classes

The last row in Figure 1 lists some supplementary guidelines to help determine the appropriate trusted process class. For example, if a trusted process supports a regrade capability by changing ONLY the sensitivity label *content* (or sensitivity level), it needs to over-ride the Tranquillity Principle ONLY and is, therefore, a Class 4 Trusted Process. In a similar fashion, a trusted process supporting a program producing an unclassified report about the existence of some classified fact, but not the value of the fact, needs to over-ride MAC enforcement by examining the content of some object -- a Class 2 or Class 3 Trusted Process.

The right-most column in Figure 1 gives the potential operations a trusted process class might perform. Thus, a Class 4 Trusted Process may have to both read and write to the sensitivity label for an object (or subject) as it operates with Tranquillity Principle over-ride privileges. Note the Class 0 Trusted Processes are special cases which require no over-ride privilege, but are, accordingly, restricted to *read only* operations -- as would be needed to translate the binary sensitivity label content used in the TCB-equipped operating system itself to the binary sensitivity label content used in the TCB-equipped database management system.

Figure 1 also illustrates that, as the Trusted Process Class *number* increases, so do the risks associated with using trusted processes in that class. This is especially true for Trusted Process Classes 6 and 7 where, unless there are compelling mission satisfaction reasons involved, the risks may be unacceptably high.

Finally, using the *Integrity Principle*, it is easy to show that as the Trusted Process Class *number* increases, the inherent *trustworthiness* of each class decreases since the potential for "abuse" increases. Thus, trusted processes which do not over-ride the *Tranquillity Principle* are more trustworthy than those which do.

5. **Trusted Process Implementation**. In the text above, the phrases "*... the potential operations a trusted process class might have to perform*." and "*... may have to both read and write to the sensitivity label ...*" are, regrettably, ambiguous. Resolving such ambiguity demands that the *Least Privilege Principle* be explicitly invoked for *each* and *every* trusted process. Moreover, there are MANDATORY programming confinement rules that must be carefully followed.

a. Trusted Process Programming Confinement Rules (PCR). This section cites the programming confinement rules that a trusted process developer MUST obey.

(1) Local Domain Context Storage [PCR-1]. *Each* and *every* trusted process shall use local domain context storage ONLY for variables used in the trusted process.

(2) Local Domain Context Storage Purge [PCR-2]. For *each* and *every* trusted process, the last step prior to exiting from the trusted process shall purge (i.e., set to all binary zero) *all* variables used in the trusted process.

(3) Trusted Process Audit [PCR-3]. *Each* and *every* trusted process invocation shall be auditable[3] by the TCB-equipped operating system per the schedule in Table PCR-1.

Table PCR-1, Trusted Process Audit Requirements

Trusted Process Class	Audit Required	Remarks
0	Optional	May be selectively audited
1	Optional	May be selectively audited
2	Optional for *Read* *ALWAYS for Write*	---
3	Optional for *Read* *ALWAYS for Write*	---
4	ALWAYS	No Exceptions Permitted
5	ALWAYS	No Exceptions Permitted
6	ALWAYS	No Exceptions Permitted
7	ALWAYS	No Exceptions Permitted

(4) Assignment Statement Restrictions [PCR-4]. For MAC over-ride privileged trusted processes, the *value of the object read* shall NEVER appear **ALONE** on the right-hand side of an expression in an assignment statement.

(5) Function or Subroutine Return Parameter Restrictions [PCR-5]. For MAC over-ride privileged trusted processes, the *value of the object read* shall NEVER be used as the return value for the trusted process function or subroutine.

(6) Least Privilege Principle Restrictions [PCR-6]. *Each* and *every* trusted process shall ONLY use the MINIMUM privilege(s) from the over-ridden set to perform its function.

(7) Computational Expression Restrictions [PCR-7]. For MAC over-ride privileged trusted processes, the *value of the object read* may be used in a computational expression provided that the *value of the object read* shall NEVER be revealed to any entity outside the local domain of the trusted process itself.

(8) Logical Expression Restrictions [PCR-8]. For MAC over-ride privileged trusted processes, the *value of the object read* may be used in a logical expression provided that the *value of the object read* shall NEVER be revealed to any entity outside the local domain of the trusted process itself.

(9) Single Functionality Restrictions [PCR-9]. *Each* and *every* trusted process shall perform a SINGLE well-defined function.

(10) Single Entry Restrictions [PCR-10]. *Each* and *every* trusted process shall have a SINGLE well-defined entry point for execution to begin.

(11) Single Exit Restrictions [PCR-11]. *Each* and *every* trusted process shall have a SINGLE well-defined exit point for execution to conclude.

(12) Trusted Process Author [PCR-12]. *Each* and *every* trusted process shall have an appropriately cleared assigned author whose work shall be INDEPENDENTLY verified by an appropriately cleared analyst (e.g., for a SECRET system, a SECRET cleared person).

(13) Configuration Management Handling Restrictions [PCR-13]. *Each* and *every* trusted process shall be assigned to a "Trusted Process Library" with access restricted to specifically named persons ONLY.

(14) Trusted Process Qualification Testing [PCR-14]. *Each* and *every* trusted process shall be tested comprehensively and the test results explicitly addressed in the *Security Test and Evaluation (ST&E) Report*.

b. <u>Trusted Process Programming Confinement Rule Assignment Schedule</u>. Table PCR-2 gives the schedule for applying the MANDATORY programming confinement rules cited in this section.

Table PCR-2, Trusted Process Programming Confinement Rule Assignments

	Rule Assignment(s)			
Trusted Process Class	Tranquillity Principle	Mandatory Access Controls	Discretionary Access Controls	Universally Applicable PCRs
0				1, 2, 3, 6, 9, 10, 11, 12, 13, 14
1				
2		4, 5, 7 ,8		
3		4, 5, 7, 8		
4				
5				
6		4, 5, 7, 8		
7		4, 5, 7, 8		
	Label	*Content*	*Privileges*	

6. **Summary**. The need for trusted processes appears to circumvent the basic philosophy for using a TCB-equipped operating system in the first place -- why expend valuable resources for a secure *Trusted System* and then permit "deviations" to occur with trusted processes?

The facts are that developers <u>will</u> use trusted processes to meet mission imperatives in a TCB-enriched environment. Having done so, there is an inherent responsibility to show these trusted processes exhibit “trustworthiness” in a justifiable way.

By suggesting ways to explicitly deal with “trustworthiness,” the methodology given in this paper fosters compliance with both *theory* and *practicality*. Moreover, its techniques can help in understanding the distinctions among the several types of trusted processes likely to be encountered in the “real world” -- some are relatively benign, others can entail serious, if not potentially catastrophic threats. The paper also suggests practical programming confinement rules which limit permitted actions a trusted process class can or should take thus providing a basis for assessing associated risks.

In a broader sense, “security bit-meisters” and system developers need to assure senior management that their investment in a secure system reflects a sound business decision. We believe the concepts developed in this paper can provide a common frame of reference for these necessary assurances.

References

[1] *DoD Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, December 1985.

[2] *A Guide to Understanding Security Modeling in Trusted Systems*, NCSC-TG-010, Version 1, October 1992.

[3] *A Guide to Understanding Audit in Trusted Systems*, NCSC-TG-001, Version 2, June 1988.

Acknowledgment

Comments provided by the referees pointed out where we needed to hone and refine several important concepts; we thank them for their valuable insight and helpful suggestions.